BLE

Maximizing BLE Throughput Part 2: Use Larger ATT MTU

November 7, 2017 | By: Punch Through

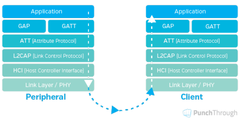
It's commonly accepted with BLE that as throughput goes up, so does power consumption. However, by intelligently choosing your MTU size, you can actually **increase throughput while also decreasing power consumption**.

NOTE: This is Part 2 in Punch Through's Maximizing BLE Throughput blog series. Here are links to our other posts in this series:

- **Part 1:** Maximizing BLE Throughput on iOS and Android
- **Part 3:** Maximizing BLE Throughput: Data Length Extension (DLE)

Journey of Data Through BLE:

Let's discuss the travel path of the application data through the Bluetooth layers, and discuss the layers that can be optimized and leveraged for throughput optimization.



Generic Attribute Profile (**GATT**) is the layer developers will often interface with the most and accustomed to. GATT defines the protocol of transferring data between two Bluetooth Low Energy devices. Application data is uniquely identified through a small entity known as an Attribute. Group of Attributes form Characteristics which adds additional properties such as permission and rules of interaction for a unique set of data. A group of Characteristics form a Large entity known as Service which adds a larger blueprint for a given feature or functionality. For Example, Battery Service includes a Battery Level Characteristic which includes the battery level of a given device.

Attribute Protocol (**ATT**) defines the protocol of transferring the attribute data. This includes GATT related functionality such as Write Request, Write Response, Notification, Read Response. In short, GATT defines and creates appropriate attributes for a given application and ATT creates outgoing and parses incoming packets that defines the action in the GATT layer.

Logical Link Control and Adaptation Protocol (**L2CAP**) is responsible for Quality of Service (QoS), routing, fragmentation and reassembly of packets for higher layer protocols such as ATT, Security Management Protocol (SMP) and etc.

Link Layer (**LL**) handles the transfer of L2CAP Packets while ensuring guaranteed delivery and integrity of data.

BLE Packet

A BLE packet containing application and transferred over the air has the following structure:

Note: The Data field is dependent on the Bluetooth specification. In Bluetooth v4.0 and 4.1, the maximum size of the Data field is **27** bytes. Bluetooth v4.2, a new feature was added to exchange the Data field length.

Closer look at the Data field

The Data field of a BLE Packet will be filled with L2CAP messages as shown below:

The L2CAP header is fixed in size (4 bytes) and placed to fulfill it's requirements such as reassembly and fragmentation of packets that are longer in length than the allowed the Data Field minus L2CAP header size of 4 bytes.

When the maximum size of Data field is **27** bytes, this allows for maximum transfer of **23** bytes of ATT data per BLE Packet.

The takeaway here is:

- Ideally, We strive to transfer **23** of application data per link layer packet or *(Link Layer Data Field Length – 4 octets)*.
- Larger application data packets of **23** *(Link Layer Data Field Length – 4 octets) or more* can be transferred from ATT->GATT->Application.

ATT MTU

ATT Maximum Transmission Unit (MTU) is the maximum length of an ATT packet. The ATT MTU is defined by the L2CAP and can be anywhere between 23 and infinity. The implementation of the Bluetooth stack is the key factor of determining the ATT MTU on both client and peripheral.

A generic ATT packet has the following structure:

Where *OP-Code* indicates the ATT Operation such as *Write Command*, *Notification*, *Read Response* or etc. The ATT Data field contains the application data.

When sending a Write, Read and Notification or Indication packets, the associated attribute handle (2 octets) will also need to be included for identification of the data.

For the default case of ATT MTU of 23 bytes, the over head bytes result in 13% of overhead data.

If you recall from the previous throughput optimization blog post [link], we chose 20 bytes of application data per BLE packet to match the default values allowed in Bluetooth v4.0 and 4.1.

ATT MTU Determination

When entering connection, client and the peripheral shall exchange their MTU through *Exchange MTU Request/Response* ATT layer command. Each side cannot transfer larger ATT packets than the other side's specified ATT_MTU.

Example

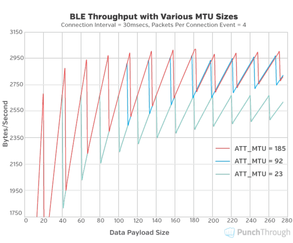
iPhone 6 and 6S ATT_MTU is 185. This results in 3/185 or 1.6% of header overhead for a payload of 182 bytes. L2CAP will transfer 185/23 or 9 Link Layer Packets.

When the ATT_MTU is 23, the payload of 185 bytes will be fragmented into 185/20 or 10 ATT packets, which results in 8 Link Layer packets.

Throughput Comparison:

Our calculation in based on 4 BLE packets in a connection event and 30 milliseconds connection interval. These are similar settings as to iOS devices.

We have graphed the throughput difference when the a given connection uses larger ATT_MTU sizes to send stream of data.



Conclusion:

When using larger ATT_MTU, the throughput is increased about 0-15% as we eliminate transferring ATT layer overhead bytes and replacing them with data. Using ATT_MTU sizes that are multiples of 23 bytes or (Link Layer Data Field – L2CAP Header Size(4 bytes)) is ideal.

As throughput increases by using larger ATT packets, this can require in creating less BLE packets which will reduce the power consumption as well.

Punch Through

Learning and collaboration efforts from current and past employees and guest authors of Punch Through.

RECENT POSTS

Meet the Team - Tyler Grunenwald
Erin Moore

Mastering all 5 Kotlin Scope Functions
Chee Yi Ong

How to Delete Bonds with the Nordic nRF5 SDK
Amanda Christiansen

Meet the Team - Matt Kruger
Emily Hinderaker

CONTACT

MINNEAPOLIS
123 N. 3rd St.
Suite 500
Minneapolis, MN
55401

info@punchthrough.com